IN THE UNITED STATES PATENT AND TRADEMARK OFFICE
BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES

Application. No: 10/051,268	§
Filed: January 18, 2002	§
Inventor(s): Sundeep Chandhoke, Nicolas Vazquez, David W Fuller and Christopher Cifra	§
Title: System and Method for Programmatically Generating a Graphical Program Based on a Sequence of Motion Control, Machine Vision, and Data Acquisition (DAQ) Operations	§
Examiner: Pham, Christine G	§
Group/Art Unit: 2192	§
Atty. Dkt. No: 5150-58300	§

(AMENDED) APPEAL BRIEF

Dear Sir or Madam:

Further to the Notice of Appeal filed February 1, 2006 and the Notice of Non-Compliant Appeal Brief mailed August 1, 2006, Appellant presents this amended Appeal Brief. Appellant respectfully requests that this appeal be considered by the Board of Patent Appeals and Interferences.

I. REAL PARTY IN INTEREST

The subject application is owned by National Instruments Corporation, a corporation organized and existing under and by virtue of the laws of the State of Delaware, and having its principal place of business at 11500 N. MoPac Expressway, Bldg. B, Austin, Texas 78759-3504.

II. RELATED APPEALS AND INTERFERENCES

No related appeals or interferences are known which would directly affect or be directly affected by or have a bearing on the Board's decision in this appeal.

III. STATUS OF CLAIMS

Claims 1 – 45 were originally filed in the application. In an amendment filed February 14, 2005, claims 2, 3, 29, and 30 were canceled. In an amendment filed July 3, 2006, claim 40 was canceled. Claims 1, 4-28, and 31-39, and 41-45 remain pending in the application. All of the pending claims stand rejected and are the subject of this appeal. A copy of the claims, as incorporating entered amendments and as on appeal, is included in the Claims Appendix hereto.

IV. STATUS OF AMENDMENTS

No amendments to the claims have been filed subsequent to the amendment of July 3, 2006. The Claims Appendix hereto reflects the current state of the claims.

V. SUMMARY OF THE CLAIMED SUBJECT MATTER

Claim 1 recites a method for creating a graphical program based on a sequence of operations that includes motion control, machine vision, and data acquisition (DAQ) operations. The method comprises displaying a graphical user interface (GUI) that provides GUI access to a set of operations, where the set of operations includes one or more motion control operations, one or more machine vision operations, and one or more DAQ operations. (*See 401 of Figure 4; Figures 6A – 6F; p. 19, line 20 – p. 21, line 5*).

The method further comprises receiving user input to the graphical user interface, where the user input specifies a desired sequence of operations. (*See 403 of Figure 4; Figure 5; Figures 6A – 6F; p. 21, lines 7-18; p. 25, line 1 – p. 27, line 7*). For example, the user may interact with the displayed graphical user interface in order to select various motion control/machine vision/DAQ operations to be included in the sequence, e.g., for a particular motion control, machine vision, and/or DAQ application. The sequence of operations specified by the user includes at least one motion control operation, at least one machine vision operation, and at least one DAQ operation.

The method further comprises storing the specified sequence of operations. (*See p. 21, lines 19-25*). For example, information representing the sequence of operations specified by the user may be stored, e.g., in a data structure representing the sequence.

The method further comprises automatically generating a graphical program to implement the specified sequence of operations. (*See 405B of Figure 4; Figure 7; Figures 8A – 8G; p. 23, lines 3-22; p. 42, lines 10-p. 48, line 7*). As well known in the prior art in the field of graphical programming, a user typically manually creates a graphical program by selecting various function nodes or icons and interconnecting them, e.g., by drawing lines or wires between them. The resulting interconnected nodes visually indicate functionality of the graphical program, e.g., visually indicate a function or process performed by the graphical program. The interconnected nodes that visually indicate the graphical program's functionality are referred to as the graphical code for the graphical program.

In the method of claim 1, the graphical program is automatically generated, e.g., in order to implement the sequence of operations specified by the user. In other words,

the graphical code for the graphical program (i.e., the plurality of interconnected nodes) is automatically generated without direct user input, e.g., as opposed to the user manually creating the graphical program by providing direct user input to create the graphical code for the graphical program.

Figures 6A – 6F may be helpful in understanding the above description of the method of claim 1. Figures 6A – 6F illustrate an exemplary graphical user interface with which a user may interact in order to specify a sequence of motion control operations. The user presses the buttons 600 (see Figure 6A) in order to add the desired motion control operations to the sequence. Each motion control operation that has been added to the sequence is represented by a respective icon 604 in the icon strip 602 (see Figure 6A). Figure 6F illustrates the sequence of motion control operations after the user has added four motion control operations, represented as four respective icons 604 in the icon strip 602. (Note that these icons are not graphical program icons, but simply represent the motion control operations that have been added to the sequence.)

Figures 8A – 8G illustrate a graphical program that has been automatically generated based on the sequence of motion control operations specified by the user in Figures 6A – 6F. (Figures 8A – 8G illustrate a single graphical program, but the size of the program requires it to be separated into multiple drawings.) Thus, the user interacts with the graphical user interface shown in Figures 6A – 6F to create a desired sequence of operations, and the graphical program of Figures 8A – 8G is automatically generated, where the graphical program is executable to perform the sequence of operations which the user specified.

Claim 28 is a memory medium claim analogous to the method claim 1 and recites similar limitations as claim 1. The memory medium comprises program instructions for creating a graphical program based on a sequence that includes motion control, machine vision, and data acquisition (DAQ) operations.

The program instructions are executable to display a graphical user interface (GUI) that provides GUI access to a set of operations, where the set of operations includes one or more motion control operations, one or more machine vision operations,

and one or more DAQ operations. (*See 401 of Figure 4; Figures 6A – 6F; p. 19, line 20 – p. 21, line 5*).

The program instructions are further executable to receive user input to the graphical user interface, where the user input specifies a desired sequence of operations. (*See 403 of Figure 4; Figure 5; Figures 6A – 6F; p. 21, lines 7-18; p. 25, line 1 – p. 27, line 7*). For example, the user may interact with the displayed graphical user interface in order to select various motion control/machine vision/DAQ operations to be included in the sequence, e.g., for a particular motion control, machine vision, and/or DAQ application. The sequence of operations specified by the user includes at least one motion control operation, at least one machine vision operation, and at least one DAQ operation.

The program instructions are further executable to store the specified sequence of operations. (*See p. 21, lines 19-25*). For example, information representing the sequence of operations specified by the user may be stored, e.g., in a data structure representing the sequence.

The program instructions are further executable to automatically generate a graphical program to implement the specified sequence of operations. (*See 405B of Figure 4; Figure 7; Figures 8A – 8G; p. 23, lines 3-22; p. 42, lines 10-p. 48, line 7*). As well known in the prior art in the field of graphical programming, a user typically manually creates a graphical program by selecting various function nodes or icons and interconnecting them, e.g., by drawing lines or wires between them. The resulting interconnected nodes visually indicate functionality of the graphical program, e.g., visually indicate a function or process performed by the graphical program. The interconnected nodes that visually indicate the graphical program's functionality are referred to as the graphical code for the graphical program.

In claim 28, the graphical program is automatically generated, e.g., in order to implement the sequence of operations specified by the user. In other words, the graphical code for the graphical program (i.e., the plurality of interconnected nodes) is automatically generated without direct user input, e.g., as opposed to the user manually creating the graphical program by providing direct user input to create the graphical code for the graphical program.

Claim 37 is a system claim analogous to the method claim 1 and recites similar limitations as claim 1. The system comprises a processor, a memory storing program instructions, and a display device. (*See computer system 82 of FIG. 1; See also CPU 180 and main memory 166 of FIG. 3*).

The processor is operable to execute the program instructions to display a graphical user interface (GUI) that provides access to a set of operations, where the set of operations includes one or more motion control operations, one or more machine vision operations, and one or more DAQ operations. (*See 401 of Figure 4; Figures 6A – 6F; p. 19, line 20 – p. 21, line 5*).

The processor is further operable to execute the program instructions to receive user input to the graphical user interface, where the user input specifies a desired sequence of operations. (*See 403 of Figure 4; Figure 5; Figures 6A – 6F; p. 21, lines 7-18; p. 25, line 1 – p. 27, line 7*). For example, the user may interact with the displayed graphical user interface in order to select various motion control/machine vision/DAQ operations to be included in the sequence, e.g., for a particular motion control, machine vision, and/or DAQ application. The sequence of operations specified by the user includes at least one motion control operation, at least one machine vision operation, and at least one DAQ operation.

The processor is further operable to execute the program instructions to store the specified sequence of operations. (*See p. 21, lines 19-25*). For example, information representing the sequence of operations specified by the user may be stored, e.g., in a data structure representing the sequence.

The processor is further operable to execute the program instructions to automatically generate a graphical program to implement the specified sequence of operations. (*See 405B of Figure 4; Figure 7; Figures 8A – 8G; p. 23, lines 3-22; p. 42, lines 10-p. 48, line 7*). As well known in the prior art in the field of graphical programming, a user typically manually creates a graphical program by selecting various function nodes or icons and interconnecting them, e.g., by drawing lines or wires between them. The resulting interconnected nodes visually indicate functionality of the graphical program, e.g., visually indicate a function or process performed by the graphical

program. The interconnected nodes that visually indicate the graphical program's functionality are referred to as the graphical code for the graphical program.

In claim 37, the graphical program is automatically generated, e.g., in order to implement the sequence of operations specified by the user. In other words, the graphical code for the graphical program (i.e., the plurality of interconnected nodes) is automatically generated without direct user input, e.g., as opposed to the user manually creating the graphical program by providing direct user input to create the graphical code for the graphical program.

Claim 41 is a method claim that recites similar limitations as claim 1. However, in this claim, the graphical user interface provides GUI access to a set of operations including one or more machine vision operations, but not necessarily motion control or DAQ operations. Also, the sequence of operations specified by the user includes at least one machine vision operation, but not necessarily motion control or DAQ operations. Otherwise, the method is substantially similar to the method of claim 1.

The method comprises displaying a graphical user interface (GUI) that provides GUI access to a set of operations, where the set of operations includes one or more machine vision operations. (*See 401 of Figure 4; Figures 6A – 6F; p. 19, line 20 – p. 21, line 5*).

The method further comprises receiving user input to the graphical user interface, where the user input specifies a desired sequence of operations. (*See 403 of Figure 4; Figure 5; Figures 6A – 6F; p. 21, lines 7-18; p. 25, line 1 – p. 27, line 7*). For example, the user may interact with the displayed graphical user interface in order to select various operations to be included in the sequence, e.g., for a particular machine vision application. The sequence of operations specified by the user includes at least one machine vision operation.

The method further comprises storing the specified sequence of operations. (*See p. 21, lines 19-25*). For example, information representing the sequence of operations specified by the user may be stored, e.g., in a data structure representing the sequence.

The method further comprises automatically generating a graphical program to implement the specified sequence of operations. (*See 405B of Figure 4; Figure 7;*

Figures 8A – 8G; p. 23, lines 3-22; p. 42, lines 10-p. 48, line 7). As well known in the prior art in the field of graphical programming, a user typically manually creates a graphical program by selecting various function nodes or icons and interconnecting them, e.g., by drawing lines or wires between them. The resulting interconnected nodes visually indicate functionality of the graphical program, e.g., visually indicate a function or process performed by the graphical program. The interconnected nodes that visually indicate the graphical program's functionality are referred to as the graphical code for the graphical program.

In the method of claim 41, the graphical program is automatically generated, e.g., in order to implement the sequence of operations specified by the user. In other words, the graphical code for the graphical program (i.e., the plurality of interconnected nodes) is automatically generated without direct user input, e.g., as opposed to the user manually creating the graphical program by providing direct user input to create the graphical code for the graphical program.

Claim 42 is a method claim that recites similar limitations as claim 1. However, in this claim, the graphical user interface provides GUI access to a set of operations including one or more data acquisition (DAQ) operations, but not necessarily machine vision or motion control operations. Also, the sequence of operations specified by the user includes at least one DAQ operation, but not necessarily machine vision or motion control operations. Otherwise, the method is substantially similar to the method of claim 1.

The method comprises displaying a graphical user interface (GUI) that provides GUI access to a set of operations, where the set of operations includes one or more data acquisition (DAQ) operations. (*See 401 of Figure 4; Figures 6A – 6F; p. 19, line 20 – p. 21, line 5*).

The method further comprises receiving user input to the graphical user interface, where the user input specifies a desired sequence of operations. (*See 403 of Figure 4; Figure 5; Figures 6A – 6F; p. 21, lines 7-18; p. 25, line 1 – p. 27, line 7*). For example, the user may interact with the displayed graphical user interface in order to select various

operations to be included in the sequence, e.g., for a particular DAQ application. The sequence of operations specified by the user includes at least one DAQ operation.

The method further comprises storing the specified sequence of operations. (*See p. 21, lines 19-25*). For example, information representing the sequence of operations specified by the user may be stored, e.g., in a data structure representing the sequence.

The method further comprises automatically generating a graphical program to implement the specified sequence of operations. (*See 405B of Figure 4; Figure 7; Figures 8A – 8G; p. 23, lines 3-22; p. 42, lines 10-p. 48, line 7*). As well known in the prior art in the field of graphical programming, a user typically manually creates a graphical program by selecting various function nodes or icons and interconnecting them, e.g., by drawing lines or wires between them. The resulting interconnected nodes visually indicate functionality of the graphical program, e.g., visually indicate a function or process performed by the graphical program. The interconnected nodes that visually indicate the graphical program's functionality are referred to as the graphical code for the graphical program.

In the method of claim 42, the graphical program is automatically generated, e.g., in order to implement the sequence of operations specified by the user. In other words, the graphical code for the graphical program (i.e., the plurality of interconnected nodes) is automatically generated without direct user input, e.g., as opposed to the user manually creating the graphical program by providing direct user input to create the graphical code for the graphical program.

Claim 43 is a method claim that recites similar limitations as claim 1. However, in this claim, the graphical user interface provides GUI access to a set of operations including one or more motion control operations and one or more machine vision operations, but not necessarily DAQ operations. Also, the sequence of operations specified by the user includes at least one motion control operation and at least one machine vision operation, but not necessarily DAQ operations. Otherwise, the method is substantially similar to the method of claim 1.

The method comprises displaying a graphical user interface (GUI) that provides GUI access to a set of operations, where the set of operations includes one or more

motion control operations and one or more machine vision operations. (*See 401 of Figure 4; Figures 6A – 6F; p. 19, line 20 – p. 21, line 5*).

The method further comprises receiving user input to the graphical user interface, where the user input specifies a desired sequence of operations. (*See 403 of Figure 4; Figure 5; Figures 6A – 6F; p. 21, lines 7-18; p. 25, line 1 – p. 27, line 7*). For example, the user may interact with the displayed graphical user interface in order to select various motion control and machine vision operations to be included in the sequence, e.g., for a particular motion control and/or machine vision application. The sequence of operations specified by the user includes at least one motion control operation and at least one machine vision operation.

The method further comprises storing the specified sequence of operations. (*See p. 21, lines 19-25*). For example, information representing the sequence of operations specified by the user may be stored, e.g., in a data structure representing the sequence.

The method further comprises automatically generating a graphical program to implement the specified sequence of operations. (*See 405B of Figure 4; Figure 7; Figures 8A – 8G; p. 23, lines 3-22; p. 42, lines 10-p. 48, line 7*). As well known in the prior art in the field of graphical programming, a user typically manually creates a graphical program by selecting various function nodes or icons and interconnecting them, e.g., by drawing lines or wires between them. The resulting interconnected nodes visually indicate functionality of the graphical program, e.g., visually indicate a function or process performed by the graphical program. The interconnected nodes that visually indicate the graphical program's functionality are referred to as the graphical code for the graphical program.

In the method of claim 43, the graphical program is automatically generated, e.g., in order to implement the sequence of operations specified by the user. In other words, the graphical code for the graphical program (i.e., the plurality of interconnected nodes) is automatically generated without direct user input, e.g., as opposed to the user manually creating the graphical program by providing direct user input to create the graphical code for the graphical program.

Claim 44 is a method claim that recites similar limitations as claim 1. However, in this claim, the graphical user interface provides GUI access to a set of operations including one or more motion control operations and one or more DAQ operations, but not necessarily machine vision operations. Also, the sequence of operations specified by the user includes at least one motion control operation and at least one DAQ operation, but not necessarily machine vision operations. Otherwise, the method is substantially similar to the method of claim 1.

The method comprises displaying a graphical user interface (GUI) that provides GUI access to a set of operations, where the set of operations includes one or more motion control operations and one or more DAQ operations. (*See 401 of Figure 4; Figures 6A – 6F; p. 19, line 20 – p. 21, line 5*).

The method further comprises receiving user input to the graphical user interface, where the user input specifies a desired sequence of operations. (*See 403 of Figure 4; Figure 5; Figures 6A – 6F; p. 21, lines 7-18; p. 25, line 1 – p. 27, line 7*). For example, the user may interact with the displayed graphical user interface in order to select various operations to be included in the sequence, e.g., for a particular motion control and/or DAQ application. The sequence of operations specified by the user includes at least one motion control operation and at least one DAQ operation.

The method further comprises storing the specified sequence of operations. (*See p. 21, lines 19-25*). For example, information representing the sequence of operations specified by the user may be stored, e.g., in a data structure representing the sequence.

The method further comprises automatically generating a graphical program to implement the specified sequence of operations. (*See 405B of Figure 4; Figure 7; Figures 8A – 8G; p. 23, lines 3-22; p. 42, lines 10-p. 48, line 7*). As well known in the prior art in the field of graphical programming, a user typically manually creates a graphical program by selecting various function nodes or icons and interconnecting them, e.g., by drawing lines or wires between them. The resulting interconnected nodes visually indicate functionality of the graphical program, e.g., visually indicate a function or process performed by the graphical program. The interconnected nodes that visually indicate the graphical program's functionality are referred to as the graphical code for the graphical program.

In the method of claim 44, the graphical program is automatically generated, e.g., in order to implement the sequence of operations specified by the user. In other words, the graphical code for the graphical program (i.e., the plurality of interconnected nodes) is automatically generated without direct user input, e.g., as opposed to the user manually creating the graphical program by providing direct user input to create the graphical code for the graphical program.

Claim 45 is a method claim that recites similar limitations as claim 1. However, in this claim, the graphical user interface provides GUI access to a set of operations including one or more machine vision operations and one or more DAQ operations, but not necessarily motion control operations. Also, the sequence of operations specified by the user includes at least one machine vision operation and at least one DAQ operation, but not necessarily motion control operations. Otherwise, the method is substantially similar to the method of claim 1.

The method comprises displaying a graphical user interface (GUI) that provides GUI access to a set of operations, where the set of operations includes one or more machine vision operations and one or more DAQ operations. (*See 401 of Figure 4; Figures 6A – 6F; p. 19, line 20 – p. 21, line 5*).

The method further comprises receiving user input to the graphical user interface, where the user input specifies a desired sequence of operations. (*See 403 of Figure 4; Figure 5; Figures 6A – 6F; p. 21, lines 7-18; p. 25, line 1 – p. 27, line 7*). For example, the user may interact with the displayed graphical user interface in order to select various operations to be included in the sequence, e.g., for a particular machine vision and/or DAQ application. The sequence of operations specified by the user includes at least one machine vision operation and at least one DAQ operation.

The method further comprises storing the specified sequence of operations. (*See p. 21, lines 19-25*). For example, information representing the sequence of operations specified by the user may be stored, e.g., in a data structure representing the sequence.

The method further comprises automatically generating a graphical program to implement the specified sequence of operations. (*See 405B of Figure 4; Figure 7; Figures 8A – 8G; p. 23, lines 3-22; p. 42, lines 10-p. 48, line 7*). As well known in the

prior art in the field of graphical programming, a user typically manually creates a graphical program by selecting various function nodes or icons and interconnecting them, e.g., by drawing lines or wires between them. The resulting interconnected nodes visually indicate functionality of the graphical program, e.g., visually indicate a function or process performed by the graphical program. The interconnected nodes that visually indicate the graphical program's functionality are referred to as the graphical code for the graphical program.

In the method of claim 45, the graphical program is automatically generated, e.g., in order to implement the sequence of operations specified by the user. In other words, the graphical code for the graphical program (i.e., the plurality of interconnected nodes) is automatically generated without direct user input, e.g., as opposed to the user manually creating the graphical program by providing direct user input to create the graphical code for the graphical program.

Claim 25 recites a method for creating a graphical program based on a prototype that includes motion control, machine vision, and data acquisition (DAQ) functionality. The method comprises displaying a graphical user interface (GUI) that provides GUI access to a set of operations. The set of operations includes one or more motion control operations, one or more machine vision operations, and one or more DAQ operations. (*See 401 of Figure 4; Figures 6A – 6F; p. 19, line 20 – p. 21, line 5*).

The method further comprises receiving user input to the graphical user interface specifying a desired sequence of operations. (*See 403 of Figure 4; Figure 5; Figures 6A – 6F; p. 21, lines 7-18; p. 25, line 1 – p. 27, line 7*). The sequence of operations comprises the prototype (*See p. 13, line 5 – p. 14, line 16*) and implements the motion control, machine vision, and DAQ functionality of the prototype.

The method further comprises automatically generating a graphical program to implement the specified sequence of operations. (*See 405B of Figure 4; Figure 7; Figures 8A – 8G; p. 23, lines 3-22; p. 42, lines 10-p. 48, line 7*). Similarly as described above with reference to claim 1, automatically generating the graphical program comprises generating graphical code in the graphical program without direct user input,

where the graphical code comprises a plurality of interconnected nodes which visually indicate the functionality of the graphical program.

Claim 26 recites a method for creating a graphical program based on a prototype that specifies motion control, machine vision, and data acquisition (DAQ) functionality. The method comprises receiving user input specifying a desired sequence of operations, where the specified sequence of operations includes at least one motion control operation, at least one machine vision operation, and at least one data acquisition operation. (*See 403 of Figure 4; Figure 5; Figures 6A – 6F; p. 21, lines 7-18; p. 25, line 1 – p. 27, line 7*).

The method further comprises recording the specified sequence of operations in a data structure. The specified sequence of operations comprises the prototype. (*See p. 21, lines 19-25; p. 13, line 5 – p. 14, line 16*).

The method further comprises automatically generating a graphical program based on the prototype to implement the specified sequence of operations. (*See 405B of Figure 4; Figure 7; Figures 8A – 8G; p. 23, lines 3-22; p. 42, lines 10-p. 48, line 7*). Similarly as described above with reference to claim 1, automatically generating the graphical program comprises generating graphical code in the graphical program without direct user input, where the graphical code comprises a plurality of interconnected nodes which visually indicate the functionality of the graphical program.

Claim 39 is a system claim that recites similar limitations as claim 1. The system comprises means for displaying a graphical user interface (GUI) that provides GUI access to a set of operations, where the set of operations includes one or more motion control operations, one or more machine vision operations, and one or more DAQ operations. (*See Figure 1; 401 of Figure 4; Figures 6A – 6F; p. 19, line 20 – p. 21, line 5*).

The system further comprises means for receiving user input to the graphical user interface specifying the sequence of operations, where the specified sequence of operations includes at least one motion control operation, at least one machine vision operation, and at least one DAQ operation. (*See Figure 1; 403 of Figure 4; Figure 5; Figures 6A – 6F; p. 21, lines 7-18; p. 25, line 1 – p. 27, line 7*).

The system further comprises means for storing the specified sequence of operations based on the user input. (*See Figure 1; p. 21, lines 19-25*).

The system further comprises means for automatically generating a graphical program to implement the specified sequence of operations. Automatically generating the graphical program comprises generating graphical code in the graphical program without direct user input, where the graphical code comprises a plurality of interconnected nodes which visually indicate the functionality of the graphical program. (*See Figure 1; 405B of Figure 4; Figure 7; Figures 8A – 8G; p. 23, lines 3-22; p. 42, lines 10-p. 48, line 7*).

VI. GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

Claims 1, 4-11, 18-28, and 31-39, and 41-45 stand rejected under 35 U.S.C. 102(e) as being anticipated by Limondin et al. (U.S. Patent No. 6,226,783, hereinafter "Limondin").

Claims 12-17 stand rejected under 35 U.S.C. 103(a) as being unpatentable over Limondin in view of McDonald et al. (U.S. Patent No. 5,966,532, hereinafter "McDonald").

VII. ARGUMENT

Section 102(e) Rejections

Claims 1, 4-11, 18-28, and 31-39, and 41-45 stand rejected under 35 U.S.C. 102(e) as being anticipated by Limondin et al. (U.S. Patent No. 6,226,783, hereinafter "Limondin"). Appellant respectfully traverses these rejections for the following reasons. Different claims are addressed under respective subheadings.

Claims 1, 25, 26, 28, 37, 39, 41-45

Limondin relates generally to the use of a software "step program" to perform applications such as machine vision applications. Limondin teaches that steps are software objects (C++ Classes) which are combined to form the step program. They are organized in a tree or hierarchical fashion where steps can contain other steps (called children steps). Steps communicate parameters and results by using datum objects. Datum objects are software objects that provide standard types for all results found in a vision application and allow results from one step (output) to be used as inputs to other steps. Limondin teaches that steps and datums can be edited graphically by using GUI editors. (*See Col. 2, lines 59-64; Col. 4, lines 33-55*).

Appellant respectfully submits that Limondin does not teach the claimed subject matter. For example, claim 1 recites as follows:

> 1. (Previously Presented) A computer-implemented method for creating a graphical program based on a sequence that includes motion control, machine vision, and data acquisition (DAQ) operations, the method comprising:
>
> displaying a graphical user interface (GUI) that provides GUI access to a set of operations, wherein the set of operations includes one or more motion control operations, one or more machine vision operations, and one or more DAQ operations;
>
> receiving user input to the graphical user interface specifying the sequence of operations, wherein the specified sequence of operations includes at least one motion control operation, at least one machine vision operation, and at least one DAQ operation;
>
> storing the specified sequence of operations based on the user input; and
>
> automatically generating a graphical program to implement the specified sequence of operations, wherein said automatically generating the graphical program comprises generating graphical code in the graphical program without direct user input, and wherein the graphical code comprises a plurality of

interconnected nodes which visually indicate the functionality of the graphical program.

As described above, in the prior art of graphical programming, a user typically manually creates a graphical program by providing user input directly specifying the creation of the graphical code for the graphical program, e.g., directly specifying the various function nodes to include in the graphical program and directly specifying connections among the function nodes.

However, the method of claim 1 relates generally to the automatic generation of a graphical program. More particularly, the method comprises receiving user input to specify a sequence of operations. The method further comprises automatically generating a graphical program in order to implement the sequence of operations specified by the user. In other words, the graphical code for the graphical program (i.e., the plurality of interconnected nodes which visually indicate the functionality of the graphical program) is automatically generated without direct user input, e.g., as opposed to the user manually creating the graphical program by providing direct user input to create the graphical code for the graphical program.

Limondin does not teach automatically generating a graphical program, as recited in claim 1. The Examiner has equated the graphical program of claim 1 with the step program taught by Limondin. However, Limondin does not teach that the step program is automatically generated. Appellant respectfully submits that Limondin's step program is manually created with direct user input which specifies which steps should be in the step program, specifies the hierarchical arrangement of the steps, etc. Furthermore, the Examiner has even explicitly admitted that the step program is not automatically generated and that the creation of the step program is driven manually by the user. In the Office Action of June 3, 2005, the Examiner writes that, "The step program is created by having the user graphically manipulating/editing the step icons' parameters (i.e., inputs and outputs) and connections between the icons. The execution order of the steps can also be graphically defined by the user." Therefore, Limondin's step program is created in response to direct user input. Limondin teaches nothing at all about automatically generating the step program or generating "graphical code" in the step program without direct user input.

Furthermore, Appellant notes that the Examiner has asserted that the "graphical program" of claim 1 and the "sequence of operations" of claim 1 are both equivalent to Limondin's step program. However, the sequence of operations specified by the user, and the graphical program that is automatically generated in order to implement the sequence of operations, are two different things. For example, Figures 6A – 6F illustrate an exemplary graphical user interface with which the user may interact in order to specify a sequence of motion control operations. The user presses the buttons 600 (see Figure 6A) in order to add the desired motion control operations to the sequence. Each motion control operation that has been added to the sequence is represented by a respective icon 604 in the icon strip 602 (see Figure 6A). Figure 6F illustrates the sequence of motion control operations after the user has added four motion control operations, represented as four respective icons 604 in the icon strip 602. (Note that these icons are not graphical program icons, but simply represent the motion control operations that have been added to the sequence.)

Figures 8A – 8G illustrate a graphical program that has been automatically generated based on the sequence of motion control operations specified by the user in Figures 6A – 6F. (Figures 8A – 8G illustrate a single graphical program, but the size of the program requires it to be separated into multiple drawings.) Thus, the user interacts with the graphical user interface shown in Figures 6A – 6F to create a desired sequence of operations, and the graphical program of Figures 8A – 8G is automatically generated, where the graphical program is executable to perform the sequence of operations which the user specified.

Thus, the sequence of operations specified by the user, and the graphical program that is automatically generated in order to implement the sequence of operations, are two different things. However, the Examiner has attempted to equate both the sequence of operations and the graphical program to the same thing, i.e., Limondin's step program. Appellant respectfully argues that Limondin does not teach the subject matter of receiving user input to specify a sequence of operations and automatically generating a graphical program in order to implement the specified sequence of operations, i.e., automatically generating a graphical program based on the sequence of operations specified by the user.

Thus, for at least the reasons provided above, Appellant respectfully submits that Limondin does not teach or suggest the subject matter of claim 1, and thus, claim 1 is patentably distinct over Limondin. Inasmuch as the independent claims 25, 28, 37, 39-45 recite similar limitations as claim 1, Appellant respectfully submits that these independent claims are also patentably distinct over Limondin, for reasons similar to those set forth above with respect to claim 1.

As per independent claim 26, the claim recites as follows:

> 26. (Previously Presented) A computer-implemented method for creating a graphical program based on a prototype that specifies motion control, machine vision, and data acquisition (DAQ) functionality, the method comprising:
>
> receiving user input specifying a desired sequence of operations, wherein the specified sequence of operations includes at least one motion control operation, at least one machine vision operation, and at least one data acquisition operation;
>
> recording the specified sequence of operations in a data structure, wherein the specified sequence of operations comprises the prototype; and
>
> automatically generating a graphical program based on the prototype to implement the specified sequence of operations, wherein said automatically generating the graphical program comprises generating graphical code in the graphical program without direct user input, and wherein the graphical code comprises a plurality of interconnected nodes which visually indicate the functionality of the graphical program.

For reasons similar to those discussed above with respect to claim 1, Appellant submits that Limondin does not teach at least the limitation of, "automatically generating a graphical program based on the prototype". Appellant also notes that the Examiner does not give any reasoning for the rejection of claim 26, and it is not clear what in Limondin the Examiner considers to be the prototype. Appellant assumes that the Examiner has again equated both the prototype and the graphical program with Limondin's step program. However, the prototype and the graphical program are not the same thing. (As recited in the claim, the graphical program is automatically generated based on the prototype.)

Appellant thus respectfully submits that all of the independent claims are patentably distinct over the cited art. For at least this reason, Appellant also respectfully submits that the dependent claims are also allowable over the cited art. Appellant also

respectfully submits that numerous ones of the dependent claims recite further distinctions over the cited art, as discussed below.

Appellant also notes that claims 1, 25, 26, 28, 37, 39, 42, 44, and 45 recite additional limitations regarding data acquisition (DAQ) operations. For example, claim 1 recites in pertinent part, "displaying a graphical user interface (GUI) that provides GUI access to… one or more DAQ operations" and "wherein the specified sequence of operations includes… at least one DAQ operation". Appellant respectfully submits that Limondin does not teach these limitations.

The Field of the Invention of the present application states that, "The present invention also relates to the fields of computer-based motion control, computer-based machine vision, and computer-based data acquisition (DAQ)." As well known to those skilled in the art of test and measurement applications, the "DAQ" acronym is not simply an abbreviation for the words "data acquisition", but refers to specific instrumentation technology for acquiring measurement data. Appellant notes that Limondin does not teach the use of DAQ operations or DAQ measurement devices. In fact, the term "DAQ" is entirely absent from Limondin's disclosure.

As the Board is certainly aware, anticipation under a Section 102(e) rejection requires that, "[t]he identical invention must be shown in as complete detail as is contained in the patent claim. *Jamesbury Corp.,* 756 F.2d at 1560, 225 USPQ at 256; *Connell,* 722 F.2d at 1548, 220 USPQ at 198." *Richardson v. Suzuki Motor Co.*, 9 USPQ2d 1913, 1920 (Fed. Cir. 1989). Limondin does not teach the limitations of, "displaying a graphical user interface (GUI) that provides GUI access to… one or more DAQ operations" and "wherein the specified sequence of operations includes… at least one DAQ operation". Appellant thus respectfully submits that claims 1, 25, 26, 28, 37, 39, 42, 44, and 45 are also patentably distinct over Limondin for this further reason.

Claim 5

Claim 5 recites the additional limitations of, "wherein the graphical program includes a block diagram portion and a user interface panel portion." As known in the art of graphical programming, the block diagram of a graphical program typically includes a plurality of function nodes that are interconnected in order to define and visually indicate

the functionality of the graphical program. The graphical program may also have a user interface panel portion for receiving user input to and/or for displaying output from the graphical program during its execution.

The Examiner has equated the "graphical program" of claims 1 and 5 with Limondin's step program. However, Limondin does not teach that the step program comprises both a block diagram portion and a user interface panel portion. With respect to the user interface panel portion, the Examiner states, "e.g., see *step program*, *tree control window* col. 4:50-55; see FIG. 2 and associated text". However, with respect to the tree control window and FIG. 2, Limondin teaches that, "FIG. 2 shows the editor object that is created to display an entire step program graphically in a tree control window." Appellant respectfully submits that the tree control window refers to a window useable for editing or creating a step program. However, the tree control window is not a user interface panel portion of the step program, i.e., the step program does not comprise the tree control window.

Limondin teaches nothing whatsoever about a step program comprising both a block diagram portion and a user interface panel portion. In fact, Appellant cannot find any teaching at all in Limondin about a user interface panel portion of a step program, either for receiving user input to the step program during its execution or for displaying output from the step program.

Appellant thus submits that Limondin does not teach the limitations recited in claim 5.

Claims 6 and 31

Appellant notes that the Examiner provides no reasoning for the rejection of claims 6 and 31. Claims 6 and 31 recite additional limitations not taught by Limondin. For example, claim 6 recites the additional limitation of, "wherein the graphical program is a graphical data flow program." As well known in the art of graphical programming, the term "graphical data flow program" refers to a graphical program where the nodes are connected according to a data flow format, e.g., such that nodes in the graphical program are interconnected by lines that visually indicate data flow among the nodes. For example, a line or wire from an output port of a first node to an input port of a second

node may serve as a visual indication that output data produced by the first node is passed as input data to the second node.

Appellant respectfully submits that Limondin's step program is not a graphical data flow program and does not visually indicate data flow among the nodes. In FIG. 2, Limondin illustrates a step program arranged as a tree hierarchy. Appellant respectfully submits that data flow among the steps in the step program is not clear from the tree hierarchy. Consider the "FindPin2" step, for example. It is not clear from the tree hierarchy whether the "FindPin2" step receives data from any of the previous steps or whether it passes data to any of the subsequent steps. Furthermore, a tree hierarchy such as shown in FIG. 2 is a fairly rigid structure that does not allow data flow among the steps in the step program to be uniquely represented. For example, suppose that in one case the "FindPin2" step only passes data directly to the "WarpStep" step and not to the "I/O Output" step. Suppose that in another case the "FindPin2" step passes data directly to both the "WarpStep" step and to the "I/O Output" step. In both of these cases, each of these steps would apparently be displayed in a tree structure such as taught in Limondin in exactly the same way. Limondin does not teach the concept of a graphical data flow program suitable for visually indicating the particular data flow that occurs among a plurality of nodes.

Appellant thus submits that Limondin does not teach the limitations recited in claims 6 and 31.

Claim 7

Claim 7 recites the additional limitation of, "wherein said programmatically generating the graphical program comprises including one or more nodes in the graphical program corresponding to the operations in the sequence." Appellant notes that, as per claim 1, the graphical program is automatically generated such that the nodes in the graphical program are automatically included without direct user input. As discussed above with respect to claim 1, Limondin teaches the manual creation of a step program in response to direct user input, but does not teach automatically generating a graphical program. More particularly, Limondin does not teach automatically including one or

more nodes in a graphical program, where the one or more nodes correspond to operations in a sequence of operations specified by user input.

Appellant thus submits that Limondin does not teach the limitations recited in claim 7.

Claim 8

Appellant notes that the Examiner provides no reasoning for the rejection of claim 8. Claim 8 recites the additional limitations of,

> "wherein said programmatically generating the graphical program comprises:
> generating portions of graphical code, wherein each portion of graphical code implements one of the operations in the sequence; and
> linking the portions of graphical code together."

Appellant notes that, as per claim 1, graphical code is automatically generated in the graphical program without direct user input. Limondin teaches the manual creation of a step program in response to direct user input, but does not teach automatically generating a graphical program. More particularly, Limondin does not teach the automatic generation of a graphical program which comprises 1) generating portions of graphical code, wherein each portion of graphical code implements one of the operations in a sequence of operations that has been specified by user input; and 2) linking the portions of graphical code together.

Appellant thus submits that Limondin does not teach the limitations recited in claim 8.

Claim 9

Claim 9 recites the additional limitations of,

> "wherein each portion of graphical code includes one or more graphical program nodes, wherein each node has one or more inputs or outputs;
> wherein generating each portion of graphical code comprises connecting the node inputs and outputs together in order to implement the operation with which the portion of graphical code is associated."

As discussed above with respect to claim 8, Limondin does not teach the automatic generation of a graphical program, where this comprises generating portions of graphical code, wherein each portion of graphical code implements one of the operations in a sequence of operations that has been specified by user input. Limondin also does not teach the further limitation of, "wherein generating each portion of graphical code comprises connecting the node inputs and outputs together in order to implement the operation with which the portion of graphical code is associated."

Appellant thus submits that Limondin does not teach the limitations recited in claim 9.

Claim 10

Claim 10 recites the additional limitations of, "wherein linking a first portion of graphical code to a second portion of graphical code comprises connecting an output of a node in the first portion of graphical code to an input of a node in the second portion of graphical code." As discussed above with respect to claim 8, Limondin does not teach the automatic generation of a graphical program, where this comprises generating portions of graphical code, wherein each portion of graphical code implements one of the operations in a sequence of operations that has been specified by user input. Limondin also does not teach the concept of linking a first portion of graphical code to a second portion of graphical code by connecting an output of a node in the first portion of graphical code to an input of a node in the second portion of graphical code.

Appellant thus submits that Limondin does not teach the limitations recited in claim 10.

Claim 11

Claim 11 recites the additional limitations of,

> "for each operation in the sequence, retrieving information associated with the operation from a database;
>
> wherein generating the portion of graphical code that implements a particular operation utilizes the database information retrieved for the particular operation."

In the rejection of claim 11, the Examiner states, "see recipe database col. 2:1-6; see GUID, database col. 4:55-65". As per the "recipe database," the only mention that Appellant can find of this database is at Col. 2, lines 1-6, which simply states, "3) Supports building the programs from a recipe database or directly using an easy to use point and click graphical user interface." Appellant respectfully submits that this does not amount to a clear teaching or suggestion of the particular limitations recited in claim 11. Limondin does not teach that the recipe database includes information regarding particular operations and also does not teach generating a portion of graphical code that implements a particular operation in a user-specified sequence of operations, where the generating of the portion of graphical code utilizes information retrieved for the particular operation from the recipe database.

As per the other cited portion at Col. 4, lines 55-65, Limondin simply teaches that, "In addition, each step and datum in a step program contains a special identifier or GUID (Globally Unique IDentifier) which allows the step or datum to dynamically create the editor objects that let a user change parameters and settings and train a step or datum. The GUID is stored on the host in a special database sometimes referred to as a 'registry' in the Microsoft Windows operating system, as well as inside the step and datum object itself." Appellant respectfully submits that storing the GUIDs of the steps and datums in the step program is not at all the same as performing the particular limitations recited in claim 11.

Appellant thus submits that Limondin does not teach the limitations recited in claim 11.

Claims 21 and 36

Claims 21 and 36 recite additional limitations not taught by Limondin. For example, claim 21 recites the additional limitations:

> "wherein the sequence is operable to:
> control motion of a device;
> analyze acquired images; and
> acquire measurement data."

Appellant can find no teaching in Limondin regarding a sequence of operations that is operable to perform all three of the above. Appellant thus submits that Limondin does not teach the limitations recited in claims 21 and 36.

Claim 38

Claim 38 recites additional limitations not taught by Limondin, as follows:

> 38. (Original) The system of claim 37, further comprising:
> a motion control device;
> an image acquisition device; and
> a data acquisition device;
> wherein the processor is operable to execute the graphical program to:
> control the motion control device to move an object;
> control the image acquisition device to acquire one or more images of the object; and
> control the data acquisition device to acquire measurement data of the object.

As discussed above with reference to claims 1, 25, 26, 28, 37, 39, 42, 44, and 45, Limondin contains no teaching regarding DAQ operations or the use of a DAQ device to acquire measurement data of an object.

Furthermore, Limondin does not teach a system that includes all three of: 1) a motion control device; 2) an image acquisition device; and 3) a data acquisition device.

Furthermore, Limondin does not teach a program which is executable to perform the particular application recited in claim 38, i.e., does not teach a program which is executable to perform all three of: 1) controlling a motion control device to move an object; 2) controlling an image acquisition device to acquire one or more images of the object; and 3) controlling a data acquisition device to acquire measurement data of the object.

Appellant thus submits that Limondin does not teach the limitations recited in claim 38.

Claim 24

Claim 24 recites the additional limitations of, "receiving user input to the graphical user interface for configuring one or more of the operations in the sequence"

[recited in claim 22] and "for each operation to be configured, displaying a graphical panel including graphical user interface elements for setting properties of the operation and receiving user input to the graphical panel to set one or more properties of the operation."

In the rejection of claim 24, the Examiner cites the following portions of Limondin's teaching. 1) Col. 2, lines 15-22:

> "7) Provides a mechanism for the machine vision program to create the user interface components that allow the training and setting of parameters using a GUI (Graphical User Interface) library native to the platform (computer or ystem) that the program is loaded on."

Col. 3, lines 1-15:

> "A step program encodes a wealth of information. For example, a step program:
> a) contains a list of operations that together make up the machine vision application;
> b) for each step (operation), encodes the set of parameters and settings required to execute that step (operation) successfully;
> c) for each step (operation), defines the inputs that the step (operation) accepts and the outputs or results that are generated;
> d) for outputs that are generated, attaches a system of coordinate information so that results can be expressed in real physical units;
> e) contains information that allows for the calibration of a system of coordinates to physical, real-world units like inches or mm and also allows the conversion of results from one system of coordinates to another;"

Col. 4, lines 55-65:

> "In addition, each step and datum in a step program contains a special identifier or GUID (Globally Unique IDentifier) which allows the step or datum to dynamically create the editor objects that let a user change parameters and settings and train a step or datum. The GUID is stored on the host in a special database sometimes referred to as a "registry" in the Microsoft Windows operating system, as well as inside the step and datum object itself."

The Examiner has apparently equated the operations in the sequence with the steps in Limondin's step program. However, none of the cited portions of Limondin support the Examiner's rejection of claim 24. As noted above, claim 24 recites, "for each operation to be configured, displaying a graphical panel including graphical user interface elements for setting properties of the operation and receiving user input to the graphical

panel to set one or more properties of the operation." However, Limondin does not teach the concept of displaying a graphical panel that includes graphical user interface elements for setting properties of a step in the step program and receiving user input to the graphical panel to set one or more properties of the step.

Appellant thus submits that Limondin does not teach the limitations recited in claim 24.

Section 103 Rejections

Claims 12-17 stand rejected under 35 U.S.C. 103(a) as being unpatentable over Limondin in view of McDonald et al. (U.S. Patent No. 5,966,532, hereinafter "McDonald"). Appellant respectfully traverses these rejections for the following reasons.

As discussed above with respect to claim 1, the method comprises, in part, receiving user input to specify a sequence of operations, and also comprises automatically generating a graphical program in order to implement the sequence of operations specified by the user. Claims 12-17 relate to creating an association between the sequence and the graphical program.

The Examiner asserts that McDonald teaches the concept of creating an association between a sequence of operations specified by a user and a graphical program. Appellant respectfully disagrees. McDonald relates generally to a system and method for automatically generating graphical program code based on user interface configurations created by a user (See Col. 1, lines 37-40). McDonald teaches that,

> "When the graphical code generation wizard is invoked, the wizard displays on the screen a configuration panel or dialog, prompting the user to configure the control or object. The user then selects parameter values to configure certain aspects of the graphical code being created. The graphical code generation wizard selects a graphical code template in response to the control and configures the graphical code template with the parameter values. The graphical code generation wizard then creates an association between the control and the configured graphical code. The user can edit wizard created code either using the graphical code generation wizard or by unlocking the association between the control and the code and making the changes directly in the block diagram." (*See Abstract; Col. 5, line 12 – Col. 6, line 15* (Emphasis added))

Thus, McDonald teaches creating an association between graphical code and a control, e.g., a user interface control or front panel control. McDonald does not,

however, teach creating an association between a sequence of operations specified by a user and a graphical program automatically generated based on the sequence of operations, as recited in claims 12-17.

Furthermore, even if McDonald did teach the limitation of creating an association between the sequence of operations and the automatically generated graphical program, there would be no reason to combine this feature of McDonald with Limondin. As noted above, the Examiner has equated both the sequence of operations and the graphical program with the same thing, i.e., Limondin's step program. Thus, the question arises as to what two things the association would be between in Limondin and what purpose the association would serve.

Thus, for at least the reasons set forth above, Appellant respectfully submits that Limondin and McDonald, taken either singly or in combination, do not teach the limitation recited in claims 12 and 14 of, "creating an association between the sequence and the graphical program." Appellant thus submits that for at least this reason, the Examiner has not established a case of prima facie obviousness with respect to claims 12 and 14 (and thus also claims 13 and 15-17, which depend on claims 12 and 14, respectively).

Claim 12 also recites the additional limitations of,

> "modifying the sequence to create a new sequence in response to user input after said creating the association; and
> modifying the graphical program according to the new sequence to create a new graphical program."

Appellant respectfully submits that these additional limitations are also not taught by the cited references.

Claim 13 also recites the additional limitations of,

> "wherein said modifying the graphical program according to the new sequence uses the association between the sequence and the graphical program;
> wherein the association remains between the new sequence and the new graphical program."

Appellant respectfully submits that these additional limitations are also not taught by the cited references.

Claim 17 also recites the additional limitations of,

"modifying the graphical program in response to user input after said generating the graphical program and after said creating the association between the sequence and the graphical program;

determining if an association exists between the sequence and the graphical program in response to said modifying the graphical program; and

removing the association between the sequence and the graphical program in response to said modifying."

Appellant respectfully submits that these additional limitations are also not taught by the cited references.

VIII. CONCLUSION

For the foregoing reasons, it is submitted that the Examiner's rejection of claims 1, 4-28, and 31-39, and 41-45 was erroneous, and reversal of the Examiner's decision is respectfully requested.

The Commissioner is authorized to charge the appeal brief fee of $500.00 and any other fees that may be due to Meyertons, Hood, Kivlin, Kowert, & Goetzel, P.C. Deposit Account No. 501505/5150-58300/JCH. This Appeal Brief is submitted with a return receipt postcard.

Respectfully submitted,

/Jeffrey C. Hood/
Jeffrey C. Hood, Reg. #35198
ATTORNEY FOR APPLICANT(S)

Meyertons Hood Kivlin Kowert & Goetzel, P.C.
P.O. Box 398
Austin, TX 78767-0398
Phone: (512) 853-8800

Date: September 1, 2006 JCH/JLB

IX. CLAIMS APPENDIX

The following lists the claims as incorporating entered amendments and as on appeal.

1. (Previously Presented) A computer-implemented method for creating a graphical program based on a sequence that includes motion control, machine vision, and data acquisition (DAQ) operations, the method comprising:

displaying a graphical user interface (GUI) that provides GUI access to a set of operations, wherein the set of operations includes one or more motion control operations, one or more machine vision operations, and one or more DAQ operations;

receiving user input to the graphical user interface specifying the sequence of operations, wherein the specified sequence of operations includes at least one motion control operation, at least one machine vision operation, and at least one DAQ operation;

storing the specified sequence of operations based on the user input; and

automatically generating a graphical program to implement the specified sequence of operations, wherein said automatically generating the graphical program comprises generating graphical code in the graphical program without direct user input, and wherein the graphical code comprises a plurality of interconnected nodes which visually indicate the functionality of the graphical program.

2-3. (Cancelled)

4. (Original) The method of claim 1, further comprising:

executing the graphical program to perform the sequence of operations.

5. (Original) The method of claim 1,

wherein the graphical program includes a block diagram portion and a user interface panel portion.

6. (Original) The method of claim 1,

wherein the graphical program is a graphical data flow program.

7. (Original) The method of claim 1,

wherein said programmatically generating the graphical program comprises including one or more nodes in the graphical program corresponding to the operations in the sequence.

8. (Original) The method of claim 1,

wherein said programmatically generating the graphical program comprises:

generating portions of graphical code, wherein each portion of graphical code implements one of the operations in the sequence; and

linking the portions of graphical code together.

9. (Original) The method of claim 8,

wherein each portion of graphical code includes one or more graphical program nodes, wherein each node has one or more inputs or outputs;

wherein generating each portion of graphical code comprises connecting the node inputs and outputs together in order to implement the operation with which the portion of graphical code is associated.

10. (Original) The method of claim 8,

wherein linking a first portion of graphical code to a second portion of graphical code comprises connecting an output of a node in the first portion of graphical code to an input of a node in the second portion of graphical code.

11. (Original) The method of claim 8, further comprising:

for each operation in the sequence, retrieving information associated with the operation from a database;

wherein generating the portion of graphical code that implements a particular operation utilizes the database information retrieved for the particular operation.

12. (Original) The method of claim 1, further comprising:

creating an association between the sequence and the graphical program;

modifying the sequence to create a new sequence in response to user input after said creating the association; and

modifying the graphical program according to the new sequence to create a new graphical program.

13. (Original) The method of claim 12,

wherein said modifying the graphical program according to the new sequence uses the association between the sequence and the graphical program;

wherein the association remains between the new sequence and the new graphical program.

14. (Original) The method of claim 1, further comprising:

creating an association between the sequence and the graphical program; and

locking the association between the sequence and the graphical program, wherein said locking prevents user editing of the graphical program.

15. (Original) The method of claim 14, further comprising:

unlocking the association between the sequence and the graphical program in response to user input after said locking;

directly changing the graphical program in response to user input after said unlocking.

16. (Original) The method of claim 15,

wherein said unlocking removes the association between the sequence and the graphical program.

17. (Original) The method of claim 14, further comprising:

modifying the graphical program in response to user input after said generating the graphical program and after said creating the association between the sequence and the graphical program;

determining if an association exists between the sequence and the graphical program in response to said modifying the graphical program; and

removing the association between the sequence and the graphical program in response to said modifying.

18. (Original) The method of claim 1,

wherein said receiving user input to the graphical user interface specifying a desired sequence of operations does not include receiving user input specifying programming language code to implement the sequence of operations.

19. (Original) The method of claim 1,

wherein the sequence is operable to perform one or more of:

control motion of a device;

analyze acquired images; and

acquire measurement data.

20. (Original) The method of claim 1,

wherein the sequence is operable to perform two or more of:

control motion of a device;

analyze acquired images; and

acquire measurement data.

21. (Original) The method of claim 1,

wherein the sequence is operable to:

control motion of a device;

analyze acquired images; and

acquire measurement data.

22. (Original) The method of claim 1, further comprising:

receiving user input to the graphical user interface for configuring one or more of the operations in the sequence;

wherein, for each operation, said configuring the operation affects an action which the operation is operable to perform.

23. (Original) The method of claim 22,

wherein said receiving user input to the graphical user interface for configuring one or more of the operations in the sequence does not include receiving user input specifying programming language code to configure the operations.

24. (Original) The method of claim 22, further comprising:

for each operation to be configured, displaying a graphical panel including graphical user interface elements for setting properties of the operation and receiving user input to the graphical panel to set one or more properties of the operation.

25. (Previously Presented) A computer-implemented method for creating a graphical program based on a prototype that includes motion control, machine vision, and data acquisition (DAQ) functionality, the method comprising:

displaying a graphical user interface (GUI) that provides GUI access to a set of operations, wherein the set of operations includes one or more motion control operations, one or more machine vision operations, and one or more DAQ operations;

receiving user input to the graphical user interface specifying a desired sequence of operations, wherein the specified sequence of operations implements the motion control, machine vision, and DAQ functionality of the prototype; and

automatically generating a graphical program to implement the specified sequence of operations, wherein said automatically generating the graphical program comprises generating graphical code in the graphical program without direct user input, and wherein the graphical code comprises a plurality of interconnected nodes which visually indicate the functionality of the graphical program.

26. (Previously Presented) A computer-implemented method for creating a graphical program based on a prototype that specifies motion control, machine vision, and data acquisition (DAQ) functionality, the method comprising:

receiving user input specifying a desired sequence of operations, wherein the specified sequence of operations includes at least one motion control operation, at least one machine vision operation, and at least one data acquisition operation;

recording the specified sequence of operations in a data structure, wherein the specified sequence of operations comprises the prototype; and

automatically generating a graphical program based on the prototype to implement the specified sequence of operations, wherein said automatically generating the graphical program comprises generating graphical code in the graphical program without direct user input, and wherein the graphical code comprises a plurality of interconnected nodes which visually indicate the functionality of the graphical program.

27. (Original) The method of claim 26, further comprising:

displaying a graphical user interface (GUI) that provides access to a set of operations, wherein the set of operations includes one or more motion control operations, one or more machine vision operations, and one or more DAQ operations;

wherein the user input is received to the graphical user interface.

28. (Previously Presented) A memory medium for creating a graphical program based on a sequence that includes motion control, machine vision, and data acquisition (DAQ) operations, the memory medium comprising program instructions executable to:

display a graphical user interface (GUI) that provides GUI access to a set of operations, wherein the set of operations includes one or more motion control operations, one or more machine vision operations, and one or more DAQ operations;

receive user input to the graphical user interface specifying the sequence of operations, wherein the specified sequence of operations includes at least one motion control operation, at least one machine vision operation, and at least one DAQ operation;

store the specified sequence of operations based on the user input; and

automatically generate a graphical program to implement the specified sequence of operations, wherein said automatically generating the graphical program comprises generating graphical code in the graphical program without direct user input, and wherein the graphical code comprises a plurality of interconnected nodes which visually indicate the functionality of the graphical program.

29-30. (Cancelled)

31. (Original) The memory medium of claim 28,

wherein the graphical program is a graphical data flow program.

32. (Original) The memory medium of claim 28,

wherein said receiving user input to the graphical user interface specifying a desired sequence of operations does not include receiving user input specifying programming language code to implement the sequence of operations.

33. (Original) The memory medium of claim 28, further comprising program instructions executable to execute the graphical program to perform the sequence of operations.

34. (Original) The memory medium of claim 33,

wherein said executing the graphical program to perform the sequence of operations comprises performing one or more of:

controlling motion of a device;

analyzing acquired images; and

acquiring measurement data.

35. (Original) The memory medium of claim 33,

wherein said executing the graphical program to perform the sequence of operations comprises performing two or more of:

controlling motion of a device;

analyzing acquired images; and

acquiring measurement data.

36. (Original) The memory medium of claim 33,

wherein said executing the graphical program to perform the sequence of operations comprises:

controlling motion of a device;

analyzing acquired images; and

acquiring measurement data.

37. (Previously Presented) A system for creating a graphical program based on a sequence that includes motion control, machine vision, and data acquisition (DAQ) operations, the system comprising:

a processor;

a memory storing program instructions; and

a display device;

wherein the processor is operable to execute the program instructions stored in the memory to:

display a graphical user interface (GUI) on the display device that provides access to a set of operations, wherein the set of operations includes one or more motion control operations, one or more machine vision operations, and one or more DAQ operations;

receive user input to the graphical user interface specifying the sequence of operations, wherein the specified sequence of operations includes at least one motion control operation, at least one machine vision operation, and at least one DAQ operation;

store the specified sequence of operations based on the user input; and

automatically generate a graphical program to implement the specified sequence of operations, wherein, in automatically generating the graphical program, the program instructions are executable to generate graphical code in the graphical program without direct user input, wherein the graphical code comprises a plurality of interconnected nodes which visually indicate the functionality of the graphical program..

38. (Original) The system of claim 37, further comprising:

a motion control device;

an image acquisition device; and

a data acquisition device;

wherein the processor is operable to execute the graphical program to:

control the motion control device to move an object;

control the image acquisition device to acquire one or more images of the object; and

control the data acquisition device to acquire measurement data of the object.

39. (Previously Presented) A system for creating a graphical program based on a sequence that includes motion control, machine vision, and data acquisition (DAQ) operations, the system comprising:

means for displaying a graphical user interface (GUI) that provides GUI access to a set of operations, wherein the set of operations includes one or more motion control operations, one or more machine vision operations, and one or more DAQ operations;

means for receiving user input to the graphical user interface specifying the sequence of operations, wherein the specified sequence of operations includes at least one motion control operation, at least one machine vision operation, and at least one DAQ operation;

means for storing the specified sequence of operations based on the user input; and

means for automatically generating a graphical program to implement the specified sequence of operations, wherein said automatically generating the graphical program comprises generating graphical code in the graphical program without direct user input, wherein the graphical code comprises a plurality of interconnected nodes which visually indicate the functionality of the graphical program.

40. (Canceled)

41. (Previously Presented) A computer-implemented method for creating a graphical program based on a sequence that includes machine vision operations, the method comprising:

displaying a graphical user interface (GUI) that provides GUI access to a set of operations, wherein the set of operations includes one or more machine vision operations;

receiving user input to the graphical user interface specifying the sequence of operations, wherein the specified sequence of operations includes at least one machine vision operation;

storing the specified sequence of operations based on the user input; and

automatically generating a graphical program to implement the specified sequence of operations, wherein said automatically generating the graphical program comprises generating graphical code in the graphical program without direct user input, and wherein the graphical code comprises a plurality of interconnected nodes which visually indicate the functionality of the graphical program.

42. (Previously Presented) A computer-implemented method for creating a graphical program based on a sequence that includes data acquisition (DAQ) operations, the method comprising:

displaying a graphical user interface (GUI) that provides GUI access to a set of operations, wherein the set of operations includes one or more DAQ operations;

receiving user input to the graphical user interface specifying the sequence of operations, wherein the specified sequence of operations includes at least one DAQ operation;

storing the specified sequence of operations based on the user input; and

automatically generating a graphical program to implement the specified sequence of operations, wherein said automatically generating the graphical program comprises generating graphical code in the graphical program without direct user input, and wherein the graphical code comprises a plurality of interconnected nodes which visually indicate the functionality of the graphical program.

43. (Previously Presented) A computer-implemented method for creating a graphical program based on a sequence that includes motion control and machine vision operations, the method comprising:

displaying a graphical user interface (GUI) that provides GUI access to a set of operations, wherein the set of operations includes one or more motion control operations and one or more machine vision operations;

receiving user input to the graphical user interface specifying the sequence of operations, wherein the specified sequence of operations includes at least one motion control operation and at least one machine vision operation;

storing the specified sequence of operations based on the user input; and

automatically generating a graphical program to implement the specified sequence of operations, wherein said automatically generating the graphical program comprises generating graphical code in the graphical program without direct user input, and wherein the graphical code comprises a plurality of interconnected nodes which visually indicate the functionality of the graphical program.

44. (Previously Presented) A computer-implemented method for creating a graphical program based on a sequence that includes motion control and data acquisition (DAQ) operations, the method comprising:

displaying a graphical user interface (GUI) that provides GUI access to a set of operations, wherein the set of operations includes one or more motion control operations and one or more DAQ operations;

receiving user input to the graphical user interface specifying the sequence of operations, wherein the specified sequence of operations includes at least one motion control operation and at least one DAQ operation;

storing the specified sequence of operations based on the user input; and

automatically generating a graphical program to implement the specified sequence of operations, wherein said automatically generating the graphical program comprises generating graphical code in the graphical program without direct user input, and wherein the graphical code comprises a plurality of interconnected nodes which visually indicate the functionality of the graphical program.

45. (Previously Presented) A computer-implemented method for creating a graphical program based on a sequence that includes machine vision and data acquisition (DAQ) operations, the method comprising:

displaying a graphical user interface (GUI) that provides GUI access to a set of operations, wherein the set of operations includes one or more machine vision operations and one or more DAQ operations;

receiving user input to the graphical user interface specifying the sequence of operations, wherein the specified sequence of operations includes at least one machine vision operation and at least one DAQ operation;

storing the specified sequence of operations based on the user input; and

automatically generating a graphical program to implement the specified sequence of operations, wherein said automatically generating the graphical program comprises generating graphical code in the graphical program without direct user input, and wherein the graphical code comprises a plurality of interconnected nodes which visually indicate the functionality of the graphical program.

X. EVIDENCE APPENDIX

No evidence submitted under 37 CFR §§ 1.130, 1.131 or 1.132 or otherwise entered by the Examiner is relied upon in this appeal.

XI. RELATED PROCEEDINGS APPENDIX

There are no related proceedings.